6 Batch Processing Commands

- Batch Processing Commands
- Stopping Batch Processing
- Parameters in Batch Files

Batch Processing Commands

Batch processing commands are special MS-DOS commands. Batch processing commands are usually only used in a batch processing file that you create. Some of these commands are used in interactive MS-DOS commands — the commands you enter directly at your keyboard to begin some task.

Batch processing files are programs that perform MS-DOS functions automatically. **.BAT** is the file extension MS-DOS reserves for batch processing files. AUTOEXEC.BAT is an example of a program that begins automatically — whenever you start or re-start your computer. AUTOEXEC.BAT is the file that MS-DOS searches for on the default drive, then sequentially performs each of the commands found in that file.

You use batch processing files to perform repetitive MS-DOS tasks. A .BAT file may contain any MS-DOS command. The special MS-DOS commands described in this chapter let you more closely control the tasks your batch processing files perform.

These commands add flexibility and power to your batch processing programs.

If you are not writing batch programs, you do not need to read this section.

Stopping Batch Processing

Batch processing files are stopped by pressing **CTRL C** or **CTRL SCR LOCK**. When you stop a batch file, the message

```
Abort batch job? (Y/N)
```

appears on your screen. If you want to stop the batch file, press **Y**. If not, press **N**, and the task the batch file is performing continues.

Parameters in Batch Files

Often you have repetitive tasks which are quite similar—perhaps differing only by the file being processed. A parameter is used to provide a file name to the .BAT file. This way, a batch file can immediately perform a task for you without having to be specially edited to match your current needs.

Thus UPDATE, a .BAT file, could be run twice weekly. On Wednesday night:

```
A>UPDATE.BAT MON.SLS TUES.SLS WEDS.SLS
```

And on Saturday night:

```
A>UPDATE.BAT THURS.SLS FRI.SLS SAT.SLS
```

Up to 10 parameters may be used. They are named %0,%1 . . . . %9. Parameter values are located on the .BAT command line. Parameters must be separated by spaces.

```
A>%0[.BAT] %1 %2 . . . . %9
```

%0 is always the name of the .BAT file.

Variables in Batch Files

Variables provide another way of passing specific data to a batch processing file.

There is an important difference between variables and parameters. A variable, once declared, retains its value until you either reset it or turn the computer off. Thus .BAT files pass values to other .BAT files and to the system.

A variable is a text string (a name) enclosed by % signs. For example, %variable%.

Values are given to variables with the MS-DOS SET command. For example, if your batch file contains the statement

```
LINK %FILE%
```

you can set the name that MS-DOS uses for that variable with the SET command. By entering

```
SET FILE = DOMORE
```

the %FILE% parameter is replaced with the filename DOMORE.

Purpose

The batch command ECHO is a toggle command. ECHO lets you choose whether or not to have the commands in your batch file displayed on your screen as each command is performed.

Syntax

ECHO [ON|OFF|<message>]

Comments

Commands in a batch file are normally displayed as they are performed by your computer. If you enter ECHO OFF into a batch file the commands are not displayed as they are performed. ECHO ON causes the commands to be displayed when they are performed.

If neither ON or OFF is specified, entering ECHO displays the current setting on your screen.

If ECHO is set to OFF, you can use ECHO <message> to display text on your screen. You use this only in a batch processing file.

FOR

Purpose

The FOR command is used to expand the power of a batch or interactive command.

Syntax

Batch processing:

```
FOR %%<c> IN <set> DO <command>
```

Interactive processing:

```
FOR %<c> IN <set> DO <command>
```

Comments

<c> is a variable parameter. <c> can be any character except 0,1,2,3...,9. This avoids confusion with the %0-%9 batch parameters.

<set> is a list of items separated by spaces and enclosed in parentheses. For example

```
(item1 item2 item3 ... itemN)
```

is a <set>.

The %%<c> variable is set sequentially to each member of <set>, and then <command> is performed. If a member of <set> is an

expression involving either * or ?, or both, then the variable is set to each matching pattern from the file.

In this case, only one such <item> may be in the set. Any <item> other than the first <item> is ignored.

Examples

```
FOR %%f IN ( *. ASM ) DO MASM %%f;

FOR %%f IN (BAK ART BUDGT) DO REM %%f
```

The '%%' is needed so that after batch parameter (%0-%9) processing is complete, a '%' remains. If only '%f' was entered, the MS-DOS batch parameter processor sees the first '%', looks at 'f', decides that '%f' is a bad parameter reference, and discards the '%f'. The FOR command would never receive this parameter. In a batch file, you must use the expression '%%'.

If you are in interactive MS-DOS processing mode, only **one** '%' is needed.

You cannot nest FOR commands in MS-DOS like a FOR-NEXT command is used in BASIC. If you try to do this, the message

```
FOR cannot be nested
```

appears when you are running your batch program. The program will not perform as you expected.

GOTO

Purpose

The GOTO command is a batch file-only command used to direct your program to perform the command following the item described in the command modifier.

Syntax

GOTO <label>

Comments

A <label> must be included in the GOTO command entry. <label> is the place in your batch file that you want the program to jump to, then perform the commands that **follow** that command.

For example,

```
:spot
REM This is a loop
GOTO spot
```

causes an infinite sequence of messages to be displayed:

```
A>REM This is a loop

A>GOTO spot

A>REM This is a loop

A>GOTO spot

A>REM This is a loop
.
.
.
etcetera
```

If you do not include <label> in the GOTO command, the batch processing file stops.

Any line in a batch processing file that begins with ':' is ignored.

IF

Purpose

The IF command allows conditional processing of MS-DOS commands.

Syntax

IF <condition> <command>

Comments

When <condition> is true, then the MS-DOS <command> is performed.

When <condition> is not true, the <command> is ignored and the next line in the batch processing file is performed.

<condition> must be one of the following:

ERRORLEVEL <number>

True if and only if the previous program executed by COMMAND had an exit code of <number> or higher.

<string1> = = <string2>

True if and only if <string1> and <string2> are identical after parameter substitution. Strings may not contain any punctuation marks that MS-DOS uses for other purposes.

EXIST <filename>

True if and only if <filename> exists.

NOT <condition>

True if and only if <condition> is false.

Examples

```
IF NOT EXIST MYFILE
ECHO Can't find file

IF NOT ERRORLEVEL 3 LINK $1,,;
```

PAUSE

Purpose

The PAUSE command lets you stop the batch processing until some action has happened.

Syntax

PAUSE [comment]

Comments

Use the PAUSE command to suspend processing of a batch file. For example, you may want to change diskettes. A PAUSE command stops your program to allow you to do so.

When the command processor encounters PAUSE, it displays:

```
Strike a key when ready . . .
```

If you press **CTRL C** or **CTRL SCR LOCK**, this prompt appears:

```
Abort batch job (Y/N)?
```

Type Y to stop the batch file and return to MS-DOS command level. You can use the PAUSE command to segment a batch file into segments that can be stopped at any appropriate point.

<comment> is optional and should be entered on the same line as PAUSE. <comment> is used to prompt — with a meaningful message — the batch file user to take some action when the file pauses.

The <comment> is displayed **before** the "Strike a key when ready . . ." message.

SHIFT

Purpose

The SHIFT command lets your batch file access more than 10 replaceable parameters or variables.

Syntax

SHIFT

Comments

MS-DOS allows 10 parameters (%0-%9) to be used by a batch file. SHIFT lets you move more parameters into a queue, discarding the first parameter.

Additional parameters (more than ten) should be on the same command line in order to be shifted into the queue.

The SHIFT command works like this:

```
if %0 = "bak"
%1 = "art"
%2 = "budgt"
%3...%9 are empty
```

then a SHIFT results in the following:

```
%0 = "bak"
%1 = "budgt"
%2...%9 are empty
```

The %0 parameter is always the name of the current batch processing file and is not affected by a SHIFT command.

If there are more than 10 parameters given on a command line, those appearing after the 10th (%9) are shifted one at a time into %9 by successive shifts.